Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Системне програмування

Лабораторна робота №11

«Програмування команд SSE у модулях на асемблері»

Виконав:

студент групи ІО-82

Шендріков Є. О.

Залікова № 8227

Перевірив Порєв В. М.

Київ – 2020

Мета

Навчитися програмувати модулі на асемблері, у яких містяться команд SSE, команди x87 FPU, а також використовувати такі модулі у проектах C++.

Завдання

1. Створити проект Visual C++ Win32 з ім'ям Lab11.

2. Написати на асемблері процедуру обчислення скалярного добутку двох векторів із використанням команд SSE. Ім'я процедури: MyDotProduct_SSE. Процедуру оформити у окремому модулі і записати файли vectsse.asm, vectsse.h. Додати файл vectsse.asm у проект.

3. Запрограмувати на асемблері процедуру обчислення скалярного добутку двох векторів на основі команд x87 FPU без використання команд SSE. Ім'я процедури: MyDotProduct_FPU. Процедуру оформити у окремому модулі і записати файли vectfpu.asm, vectfpu.h. Додати файл vectfpu.asm у проект.

4. Запрограмувати на C++ обчислення скалярного добутку тих самих векторів як звичайну функцію C++ з ім'ям MyDotProduct, яка приймає значення двох масивів і записує результат у числову перемінну (будь-яка оптимізація при компіляції повинна бути відсутня).

5. Зробити меню для вікна програми так, щоб користувач програми мав можливість викликати процедури на асемблері MyDotProduct_SSE, MyDotProduct_FPU з модулів vectsse, vectfpu, а також функцію MyDotProduct.

6. Запрограмувати вивід результатів обчислень та виміри часу виконання скалярного добутку для трьох варіантів реалізації.

7. Отримати дизасемблерний текст функції C++ MyDotProduct. Проаналізувати код дизасемблеру, порівняти з кодом на асемблері процедури MyDotProduct_FPU.

8. Зробити висновки щодо використання модулів на асемблері у програмах на мові C++ .

Мій варіант:

Кількість елементів векторів A та B має бути **N** = 40 * 25 **= 1000**

Текст програми

lab11.cpp

```
// Lab11.cpp : Defines the entry point for the application.
//

#include "stdafx.h"
#include "Lab11.h"
#include "vectsse.h"
#include "vectfpu.h"
#include "module.h"
#include <stdio.h>
#include <string>

#define MAX_LOADSTRING 100

// Global Variables:
HINSTANCE hInst;                                                    // поточний екземпляр
TCHAR szTitle[MAX_LOADSTRING];                                  // текст рядка заголовка
TCHAR szWindowClass[MAX_LOADSTRING];                  // ім’я класу головного вікна

// Форвардні оголошення функцій, включених в цей модуль коду:
ATOM                    MyRegisterClass(HINSTANCE hInstance);
BOOL                    InitInstance(HINSTANCE, int);
LRESULT CALLBACK    WndProc(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK    About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY _tWinMain(_In_ HINSTANCE hInstance,
                     _In_opt_ HINSTANCE hPrevInstance,
                     _In_ LPTSTR    lpCmdLine,
                     _In_ int       nCmdShow)
{
       UNREFERENCED_PARAMETER(hPrevInstance);
       UNREFERENCED_PARAMETER(lpCmdLine);

       // TODO: Place code here.
       MSG msg;
       HACCEL hAccelTable;

       // Ініціалізація глобальних рядків
       LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
       LoadString(hInstance, IDC_LAB11, szWindowClass, MAX_LOADSTRING);
       MyRegisterClass(hInstance);

       // Ініціалізація додатку
       if (!InitInstance (hInstance, nCmdShow))
       {
              return FALSE;
       }

       hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC_LAB11));

       // Цикл основного повідомлення:
       while (GetMessage(&msg, NULL, 0, 0))
       {
              if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
              {
                     TranslateMessage(&msg);
                     DispatchMessage(&msg);
              }
       }

       return (int) msg.wParam;
}
```

```
//
//  FUNCTION: MyRegisterClass()
//
//  PURPOSE: реєструє клас вікна.
//
//  КОМЕНТАР:
//
//  Ця функція і її використання необхідні тільки в разі, якщо потрібно, щоб даний код
//  був сумісний з системами Win32, що не мають функції RegisterClassEx
//  яка була додана в Windows 95. Виклик цієї функції важливий для того,
//  щоб додаток отримав "якісні" дрібні значки і встановив зв'язок з ними.
//
//
ATOM MyRegisterClass(HINSTANCE hInstance)
{
       WNDCLASSEX wcex;

       wcex.cbSize = sizeof(WNDCLASSEX);

       wcex.style                  = CS_HREDRAW | CS_VREDRAW;
       wcex.lpfnWndProc     = WndProc;
       wcex.cbClsExtra             = 0;
       wcex.cbWndExtra             = 0;
       wcex.hInstance              = hInstance;
       wcex.hIcon                  = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_LAB11));
       wcex.hCursor         = LoadCursor(NULL, IDC_ARROW);
       wcex.hbrBackground   = (HBRUSH)(COLOR_WINDOW+1);
       wcex.lpszMenuName    = MAKEINTRESOURCE(IDC_LAB11);
       wcex.lpszClassName   = szWindowClass;
       wcex.hIconSm         = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_SMALL));

       return RegisterClassEx(&wcex);
}

//
//   FUNCTION: InitInstance(HINSTANCE, int)
//
//   PURPOSE: зберігає обробку примірника і створює головне вікно.
//
//   КОМЕНТАРІ:
//
//   У даній функції дескриптор екземпляра зберігається в глобальній змінній, а також
//   створюється і виводиться на екран головне вікно програми.
//
//
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
   HWND hWnd;

   hInst = hInstance; // Зберегти дескриптор екземпляру в глобальній змінній

   hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
      CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

   if (!hWnd)
   {
      return FALSE;
   }

   ShowWindow(hWnd, nCmdShow);
   UpdateWindow(hWnd);

   return TRUE;
}

__declspec(align(16)) float oA[1000];
```

```
__declspec(align(16)) float oB[1000];
__declspec(align(16)) float res;
__declspec(align(16)) char TextBuf[100];

void prepare() {
	for (long i = 0; i < 1000; i++)
	{
		oA[i] = 1.0 + i;
		oB[i] = pow(-1.0, i);
	}
}


void myVectSSE(HWND hWnd)

{
	prepare();

	SYSTEMTIME st;
	long tst, ten;
	GetLocalTime(&st);
	tst = 60000 * (long)st.wMinute + 1000 * (long)st.wSecond + (long)st.wMilliseconds;

	for (long i = 0; i<1000000; i++) //повторюємо мільйон разів
	{
		MyDotProduct_SSE(&res, oB, oA, 1000);
	}

	GetLocalTime(&st);
	ten = 60000 * (long)st.wMinute + 1000 * (long)st.wSecond + (long)st.wMilliseconds -
tst;

	sprintf_s(TextBuf, "Скалярний добуток (SSE) = %f\nЧас виконання = %ld мс", res, ten);
	MessageBox(hWnd, TextBuf, "SSE", MB_OK);
}

void myVectFPU(HWND hWnd)

{
	prepare();

	SYSTEMTIME st;
	long tst, ten;
	GetLocalTime(&st);
	tst = 60000 * (long)st.wMinute + 1000 * (long)st.wSecond + (long)st.wMilliseconds;

	for (long i = 0; i<1000000; i++) //повторюємо мільйон разів
	{
		MyDotProduct_FPU(&res, oB, oA, 1000);
	}
	GetLocalTime(&st);
	ten = 60000 * (long)st.wMinute + 1000 * (long)st.wSecond + (long)st.wMilliseconds -
tst;

	sprintf_s(TextBuf, "Скалярний добуток (FPU) = %f\nЧас виконання = %ld мс", res, ten);
	MessageBox(hWnd, TextBuf, "FPU", MB_OK);

}

float MyDotProduct(float* A, float* B, long N) {
	float result = 0;
	for (long i = 0; i < N; i++) {
		result += A[i] * B[i];
	}
	return result;
}

void vectorCPP(HWND hWnd)
```

```
{
	prepare();

	SYSTEMTIME st;
	long tst, ten;
	GetLocalTime(&st);
	tst = 60000 * (long)st.wMinute + 1000 * (long)st.wSecond + (long)st.wMilliseconds;

	for (long i = 0; i<1000000; i++)
	{
		res = MyDotProduct(oA, oB, 1000);
	}

	GetLocalTime(&st);
	ten = 60000 * (long)st.wMinute + 1000 * (long)st.wSecond + (long)st.wMilliseconds -
tst;

	sprintf_s(TextBuf, "Скалярний добуток (C++) = %f\nЧас виконання = %ld мс", res, ten);
	MessageBox(hWnd, TextBuf, "C++", MB_OK);

}

//
//  FUNCTION: WndProc(HWND, UINT, WPARAM, LPARAM)
//
//  PURPOSE:  обробляє повідомлення в головному вікні.
//
//  WM_COMMAND - обробка меню програми
//  WM_PAINT - закрасити головне вікно
//  WM_DESTROY - ввести повідомлення про вихід і повернутися.
//
//
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
	int wmId, wmEvent;
	PAINTSTRUCT ps;
	HDC hdc;

	switch (message)
	{
	case WM_COMMAND:
		wmId    = LOWORD(wParam);
		wmEvent = HIWORD(wParam);
		// Parse the menu selections:
		switch (wmId)
		{
		case IDM_ABOUT:
			DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
			break;
		case IDM_EXIT:
			DestroyWindow(hWnd);
			break;
		case ID_EXECUTE_SSE:
			myVectSSE(hWnd);
			break;
		case ID_EXECUTE_FPU:
			myVectFPU(hWnd);
			break;
		case ID_EXECUTE_C:
			vectorCPP(hWnd);
			break;
		default:
			return DefWindowProc(hWnd, message, wParam, lParam);
		}
		break;
	case WM_PAINT:
		hdc = BeginPaint(hWnd, &ps);
```

```
            // TODO: Add any drawing code here...
            EndPaint(hWnd, &ps);
            break;
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
        }
        return 0;
}

// Оброблювач повідомлень для вікна "Про програму".
INT_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
        UNREFERENCED_PARAMETER(lParam);
        switch (message)
        {
        case WM_INITDIALOG:
            return (INT_PTR)TRUE;

        case WM_COMMAND:
            if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)
            {
                EndDialog(hDlg, LOWORD(wParam));
                return (INT_PTR)TRUE;
            }
            break;
        }
        return (INT_PTR)FALSE;
}
```

vectsse.asm

```
.686
.xmm
.model flat, C
.data
.code

MyDotProduct_SSE proc dest:DWORD, pB:DWORD, pA:DWORD, bits:DWORD

        mov eax, pA ; a
        mov ebx, pB ; b
        mov edi, dest ; res
        mov ecx, bits  ; n
    xorps xmm2, xmm2
        @cycle:
            movaps xmm0, [eax+4*ecx]
            movaps xmm1, [ebx+4*ecx]
            mulps xmm0, xmm1
            haddps xmm0, xmm0
            haddps xmm0, xmm0
            addps xmm2, xmm0
            sub ecx, 4
            jge @cycle

        movaps [edi], xmm2

        ret
MyDotProduct_SSE endp

        End
```

vectfpu.asm

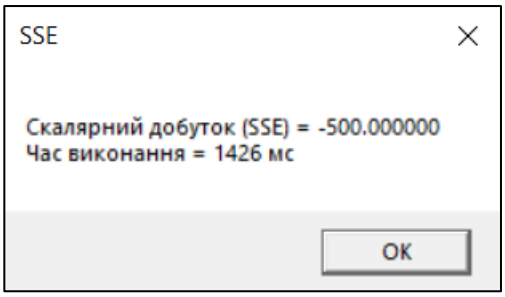
```
.586
.model flat, c
.data
.code

MyDotProduct_FPU proc dest:DWORD, pB:DWORD, pA:DWORD, bits:DWORD

       mov eax, pA ; a
       mov ebx, pB ; b
       mov edx, dest ; res
       mov ecx, bits  ; n
    dec ecx

       fldz

       @cycle:
           fld dword ptr[eax+4*ecx]
           fmul dword ptr[ebx+4*ecx]
              faddp st(1), st(0)
              dec ecx
       jge @cycle

       fstp dword ptr[edx]
       ret


MyDotProduct_FPU endp

       End
```

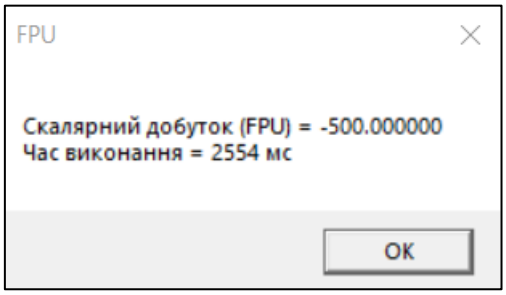
Дизасемблерний код функції MyDotProduct на C++

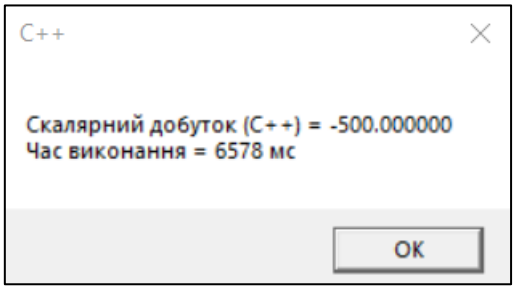
```
float MyDotProduct(float* A, float* B, long N) {
007C1AA0  push        ebp
007C1AA1  mov         ebp,esp
007C1AA3  sub         esp,0D8h
007C1AA9  push        ebx
007C1AAA  push        esi
007C1AAB  push        edi
007C1AAC  lea         edi,[ebp-0D8h]
007C1AB2  mov         ecx,36h
007C1AB7  mov         eax,0CCCCCCCCh
007C1ABC  rep stos    dword ptr es:[edi]
007C1ABE  mov         ecx,offset _F934A520_Lab11@cpp (07D0004h)
007C1AC3  call        @__CheckForDebuggerJustMyCode@4 (07C1267h)
       float result = 0;
007C1AC8  xorps       xmm0,xmm0
007C1ACB  movss       dword ptr [result],xmm0
       for (long i = 0; i < N; i++) {
007C1AD0  mov         dword ptr [ebp-14h],0
007C1AD7  jmp         MyDotProduct+42h (07C1AE2h)
007C1AD9  mov         eax,dword ptr [ebp-14h]
007C1ADC  add         eax,1
007C1ADF  mov         dword ptr [ebp-14h],eax
007C1AE2  mov         eax,dword ptr [ebp-14h]
007C1AE5  cmp         eax,dword ptr [N]
007C1AE8  jge         MyDotProduct+6Ch (07C1B0Ch)
           result += A[i] * B[i];
007C1AEA  mov         eax,dword ptr [ebp-14h]
007C1AED  mov         ecx,dword ptr [A]
007C1AF0  mov         edx,dword ptr [ebp-14h]
007C1AF3  mov         esi,dword ptr [B]
007C1AF6  movss       xmm0,dword ptr [ecx+eax*4]
007C1AFB  mulss       xmm0,dword ptr [esi+edx*4]
           result += A[i] * B[i];
```

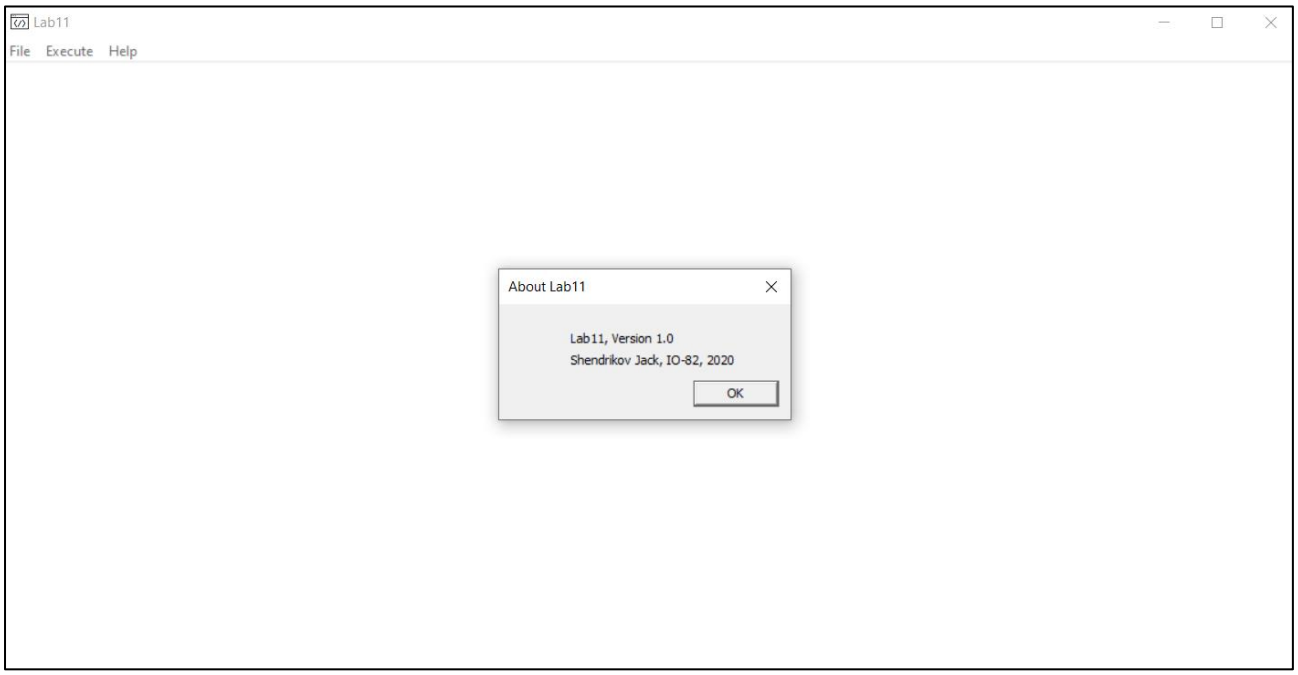
```
007C1B00  addss       xmm0,dword ptr [result]
007C1B05  movss       dword ptr [result],xmm0
      }
007C1B0A  jmp         MyDotProduct+39h (07C1AD9h)
      return result;
007C1B0C  fld         dword ptr [result]
}
007C1B0F  pop         edi
007C1B10  pop         esi
007C1B11  pop         ebx
007C1B12  add         esp,0D8h
007C1B18  cmp         ebp,esp
007C1B1A  call        __RTC_CheckEsp (07C127Bh)
007C1B1F  mov         esp,ebp
007C1B21  pop         ebp
007C1B22  ret
```

Результати роботи програми









Аналіз результатів

Програма виконує обчислення скалярного добутку двох векторів. Обчислення на мові високого рівня (C++) виявилося найбільш повільним, найшвидшим виявилося застосування команд формату SSE, що у приблизно 2 рази швидше за формат FPU та у 5 разів програмування на C++.

Результат отримано вірно для усіх способів. А для підтвердження була написана програма на мові Python, яка видала той самий результат:

```python
a, b = [], []
for i in range(1000):
    a.append(1 + i)
    b.append(pow(-1.0, i))

result = 0
for i in range(1000):
    result += a[i] * b[i]

print("Результат:", result)
```

```
Результат: -500.0
```

Отже, програма працює вірно.

Висновок

Під час виконання лабораторної роботи були покращені навички написання власних модулів, у яких містяться команди SSE та команди x87 FPU. Також навчився використовувати такі модулі у проектах C++. Було ще раз доведено швидкість виконання операцій на мові асемблеру. Кінцева мета роботи досягнута.